CLAIMS

I claim:

1. An interrupt manager for use in a distributed control system, the interrupt manager comprising:

circuitry that:

(i) receives interrupt signals including a current interrupt;

(ii) determines whether the current interrupt can be processed without delaying processing of a non-interrupt task beyond a predetermined time; and

(iii) inhibits, at least temporarily, processing of the current interrupt when it is determined that the processing of the current interrupt would delay the processing of the non-interrupt task beyond the predetermined time.

2. The interrupt manager of claim 1, wherein the circuitry determines whether the current interrupt can be processed without delaying the processing of the non-interrupt task beyond the predetermined time by determining whether a total number of interrupts including at least one of the current interrupt, a recently-performed interrupt and a pending interrupt would exceed a maximum number of interrupts.

3. The interrupt manager of claim 2, wherein the maximum number is associated with a time interval and represents a maximum number of interrupts that can be performed within the time interval.

4. The interrupt manager of claim 3, wherein the total number of interrupts includes, in addition to the current interrupt, any interrupts that have been received and not yet processed.

5. The interrupt manager of claim 3, wherein the total number of interrupts includes, in addition to the current interrupt, all interrupts that have been received since a first time.

6. The interrupt manager of claim 1, wherein the circuitry determines whether the current interrupt can be processed without delaying the processing of the non-interrupt

task beyond the predetermined time by determining whether the processing of the current interrupt could be completed within an interrupt window.

7. The interrupt manager of claim 6, wherein the interrupt window is refreshed upon expirations of window periods.

8. The interrupt manager of claim 1, wherein the circuitry determines whether the current interrupt can be processed without delaying the processing of the non-interrupt task beyond the predetermined time by determining whether the processing of the current interrupt could begin within an interrupt window.

9. The interrupt manager of claim 1, wherein the circuitry inhibits the processing of the current interrupt by masking the current interrupt.

10. The interrupt manager of claim 1, wherein the circuitry determines whether the current interrupt can be processed without delaying the processing of the non-interrupt task beyond the predetermined time by comparing a first priority associated with the current interrupt with a second priority of the non-interrupt task.

11. The interrupt manager of claim 10, wherein the first priority is that of a proxy task generated in response to the receiving of the current interrupt based upon scheduling data of a message causing the current interrupt.

12. The interrupt manager of claim 1, wherein the circuitry temporarily inhibits the processing of the current interrupt by placing information relating to the current interrupt in a later position in a queue.

13. The interrupt manager of claim 1, wherein the circuitry determines whether current interrupt can be processed without delaying the processing of the non-interrupt task beyond the predetermined time by determining whether at least one completion timing constraint associated with the non-interrupt task would be violated if the processing of the current interrupt occurred.

14. A method of handling interrupts for use with a processor in a distributed control system, the method comprising:

receiving a current interrupt signal;

determining whether processing of the current interrupt signal would delay processing of a non-interrupt task beyond a predetermined time; and

inhibiting, at least temporarily, the processing of the current interrupt signal when it is determined that the processing would delay the processing of the non-interrupt task beyond the predetermined time.

15. The method of claim 14, further comprising delaying the processing of the current interrupt signal to a later time if it is determined that the processing would delay the processing of the non-interrupt task beyond the predetermined time.

16. The method of claim 14, wherein the determining includes:

comparing a total number of interrupt signals including at least the current interrupt signal with a maximum number of interrupt signals.

17. The method of claim 14, wherein the determining includes determining whether at least one of: the processing of the current interrupt signal can be begun within a current time window; and the processing of the current interrupt signal can be completed within the current time window.

18. The method of claim 14, wherein the inhibiting occurs by at least one of: (i) masking the current interrupt signal; and (ii) placing a task related to the current interrupt signal in a queue for later processing.

19. A method of scheduling messages being transmitted on a network among spatially-distributed control components of a distributed control system, the method comprising:

receiving a message;

receiving a relative timing constraint concerning the message, wherein the relative timing constraint is indicative of an amount of time; and

inserting the message into a queue at a location that is a function of the relative timing constraint.

20. The method of claim 19, wherein the relative timing constraint is at least one of a completion timing constraint, a deadline period, and an execution period.

21. The method of claim 19, wherein the location is also a function of a priority associated with the message and of an absolute timing constraint concerning the message.

22. The method of claim 21, wherein the absolute timing constraint is a particular time.

23. The method of claim 19, wherein the inserting of the message into the queue is governed by a message scheduler implemented by a processor executing a portion of a distributed operating system providing respective portions of an overall completion timing constraint of a communication circuit to each of a plurality of application programs, the respective portions setting respective deadlines for the application programs.

24. A method of coordinating a new control application program with other control application programs being performed on a distributed real-time operating system, wherein the distributed real-time operating system is for use with a control system having spatially separated control hardware resources, the method comprising:

(a) receiving the new control application program;

(b) identifying control hardware resources from a resource list matching control hardware resources required by the new control application program;

(c) allocating portions of a constraint associated with the new control application program to each identified control hardware resource; and

(d) determining whether the allocated portions of the constraint of the new control

application program can be met while requirements of the other control application programs also are met.

25. The method of claim 24, wherein the constraint is a completion timing constraint.

26. The method of claim 24, further comprising:

collecting statistics regarding a usage of the control hardware resources as the new control application program and other control application programs are being performed; and

optimizing the usage of the control hardware resources based at least in part upon the collected statistics.

27. A method of operating an application program on a distributed control system having a plurality of hardware resources, the method comprising:

receiving high-level requirements concerning the application program;

determining low-level requirements based upon the high-level requirements;

allocating at least one of the high-level requirements and the low-level requirements among at least some of the plurality of hardware resources; and

operating the application program in accordance with the allocated requirements.

28. The method of claim 27,

wherein the high-level requirements include at least one of a hardware requirement, a completion-timing constraint, a message size, an inter-arrival period, a need for remote system services, and a type of priority, and

wherein the low-level requirements include at least one of an amount of memory, a network bandwidth, and a processor bandwidth.

29. The method of claim 27, wherein the allocating of the low-level requirements includes allocating the low-level requirements to both a primary hardware resource and an implicit hardware resource.

30. The method of claim 27, further comprising:

determining whether the allocated requirements are consistent with other allocated requirements associated with other application programs, prior to operating the application program.